基于循环神经网络的模糊测试用例生成

徐　鹏[1]，刘嘉勇[1]，林　波[1]，孙慧颖[2]，雷　斌[3]

(1. 四川大学 电子信息学院信息，成都 610065; 2. 四川省军区数据信息室，成都 610041; 3. 78100 部队，成都 610021)

摘　要：模糊测试用例的好坏是影响测试效果的重要因素。目前常规的生成方法是随机变异和人工协议分析构造，分别存在变异盲目效率低和构造复杂代价高的问题。针对上述问题提出运用深度学习技术辅助测试用例生成。利用循环神经网络处理字符文本序列的优势，通过样本数据学习训练结构特征，并预测生成符合结构特征的新数据，与随机变异算法结合构造了自动生成模型。通过以 LSTM 和 GRU 算法模型对 PDF 文件输入型测试用例生成和效果评估，生成的测试用例总体优于常规方法有较好的通过率和覆盖率。该方法通过循环神经网络的辅助实现了生成快速高效和构造难度低的优点，达到了生成效果和花费代价的平衡。

关键词：深度学习；循环神经网络；模糊测试；LSTM；GRU

中图分类号：TP399　　doi: 10.3969/j.issn.1001-3695.2018.03.0222

Generation of fuzzing test case based on recurrent neural networks

Xu Peng[1], Liu Jiayong[1], Lin Bo[1], Sun Huiying[2], Lei Bin[3]

(1. *College of Electronics & Information Engineering Sichuan University, Chengdu* 610065, *China*; 2. *Dept. of Data & Information Sichuan Provincial Military District, Chengdu* 610041, *China*; 3. *Troop* 78100, *Chengdu* 610021, *China*)

Abstract: The quality of the fuzzing test case is an important factor to affect the effectiveness of fuzzing. At present, the conventional generation method is random variation and artificial protocol analysis, which have the problems of low blind efficiency and high complexity of construction. In view of the above problems, this paper proposed the use of deep learning technology to assist test case generation. Using the advantage of recurrent neural network to deal with character text sequences, it learned training structure features through sample data, and predicted new data that conforms to structural features, and constructed an automatic generation model combined with random mutation algorithm. By using LSTM and GRU algorithm model to generate and evaluate the input type test case of PDF files, the test cases generated ware better than conventional methods with better pass rate and coverage rate. With the help of recurrent neural network, the method achieved the advantages of fast and efficient construction and low difficulty of construction, and achieves the balance of generating effect and cost.

Key words: deeplearning; recurrent neural networks; fuzzing; LSTM; GRU

0　引言

根据 2017 年 11 月美国国家标准与技术局(NIST) 发布的 NVD 个数为 1.34 万个，是 2016 年 2 倍多，出现了如勒索病毒永恒之蓝、CPU 芯片级的 Meltdown 和 Spectre 漏洞等重量级漏洞，近年来安全漏洞数量和危害程度不断上升态势。漏洞的测试和发掘已经成为信息安全的重要课题和热点。随着软件规模扩大和系统复杂度提升，模糊测试在漏洞发掘方面展现出较为优异能力和高效的水平，是漏洞挖掘中一种重要的发掘方法。经过二十余年的发展演变，模糊测试技术和方法不断改进和提升，自动化水平和适用场景都不断进步和拓展。人们针对应用场景和流程研发出诸如 PROTOS、SPIKE、FileFuzz、COMRaider、Sulley、Peach3 等更加智能、自动化的工具，极大地提高了漏洞发掘的能力和效率。但受限于信息系统种类繁多和协议、结构复杂多样等因素，其中测试用例生成技术这一重要环节仍存在一些不足：一是随机测试用例构造速度快，但存在变异的盲目性，导致通过率和代码覆盖率低，受初始样本影响大，总体效率较低；二是人工格式或协议分析对人员综合

收稿日期：2018-03-30；**修回日期：**2018-05-17

作者简介：徐鹏（1981-），男，四川仁寿人，硕士研究生，主要研究方向为信息系统安全、漏洞发掘（45221022@qq.com）；刘嘉勇（1962-），男，教授，博士，主要研究方向为信息安全理论与应用、网络通信与网络安全；林波（1984-），男，四川简阳人，硕士研究生，主要研究方向为信息系统安全；孙慧颖（1987-），女，山东海阳人，本科，主要研究方向为信息管理与信息系统；雷斌（1978-），男，陕西合阳人，硕士，工程师，主要研究方向为信息系统安全.

能力要求高，脚本编写繁琐复杂，复杂度随业务逻辑递增，时间和精力成本较高，难以复用和规模化、自动化。

得益于深度学习技术的迅猛发展和优异表现，本文提出基于循环神经网络（recurrent neural networks， RNNs）的测试用例变异生成方法，抓住测试用例是由序列化数据构成的特点，以及 RNNs 自身的结构特点是十分有利于用来建模序列型数据，将之结合到学习测试用例的序列数据，建模识别数据特征和结构并预测新产生数据,从而实现自动地生成通过率和覆盖率优异的测试用例。

1　模糊测试及测试用例生成简述

1.1　模糊测试原理和过程

模糊测试（fuzzing）是一种通过向目标系统提供非预期输入并监视异常结果来发现软件漏洞的方法，自动化地使用大量半有效的数据作为程序的输入，以程序是否出现异常作为标志，来发现应用程序中可能存在的安全漏洞。目前信息系统的种类繁多、规模和应用环境千差万别，因此测试的方法也是多种多样，但测试总体的阶段结构过程相似，如图 1 所示。

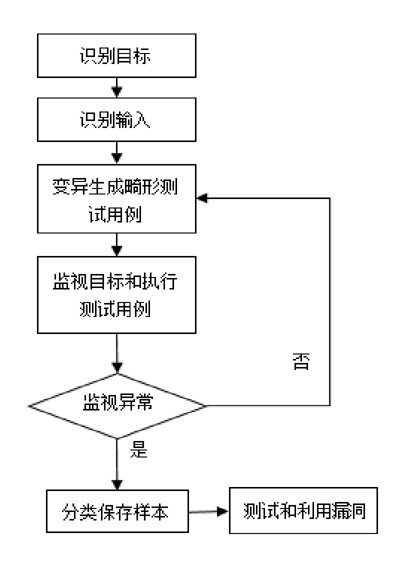


图 1　模糊测试过程

从图 1 中能够比较直观地理解，在模糊测试过程中发现漏洞的重要阶段是构造违反目标系统正常处理的测试用例，从而触发系统的异常发现系统存在的漏洞。因此，研究测试用例的生成方法是提升模糊测试效果的一项重要的内容。

1.2　测试用例生成方法简述和特点

模测试用例产生方法本质上分为基于变异的数据产生方法和基于生成的数据产生方法两类。

a)基于变异的数据生成方法是指对目标软件的输入部分有部分了解，从一个正常的样本出发，按照一定的规则和策略改变其中某些字段，从而产生新的畸形测试用例。这种方法不需要去理解当前样本文件的结构和格式，因此适用范围比较广；但对初始样本的依赖性极大，不同的初始样本会带来不同的代码覆盖率、测试深度和测试效果，所以效率较低。

b)基于生成的数据生成方法是需要对目标软件的预期输入格式非常了解，在深入研究目标软件的文件格式或协议规约的基础上，按照目标软件处理的文件格式规约或协议要求，产生违反部分规约的测试数据。这种方法产生的数据有较高的合法率及代码覆盖率，提高产生数据在校验期的通过率，覆盖可能的高危代码段，发现潜在的安全漏洞；但需要花费大量的时间和精力去完成文件格式或协议规约的理解和规则编写，不同的目标软件差异较大，难以复用、适用范围小。

测试用例生成方法也越来越多地与各种新技术和新方法结合，如遗传算法提升测试用例生成效果。当前深度学习技术在图像识别、语言处理等方面已经很优异的效果，同理在理解掌握测试用例生成原理和方法后，就自然而然地将之与深度学习技术结合起来。

2　基于循环神经网络的生成模型

2.1　循环神经网络简述

循环神经网络（recurrent neural networks，RNNs）是一种用于处理序列数据的神经网络。传统的神经网络模型是从输入层到隐含层再到输出层，层与层之间是全连接的，每层之间的节点是无连接的，这种普通的神经网络难以处理前后文关联类问题。例如，预测句子的下一个单词是什么，则需要用到前面的单词，一个句子中前后单词并不是独立的。相比一般的神经网络来说，该神经网络能够处理序列变化的数据，就能很好地解决这类问题。已经在众多自然语言处理(natural language processing, NLP)中取得了巨大成功以及广泛应用。

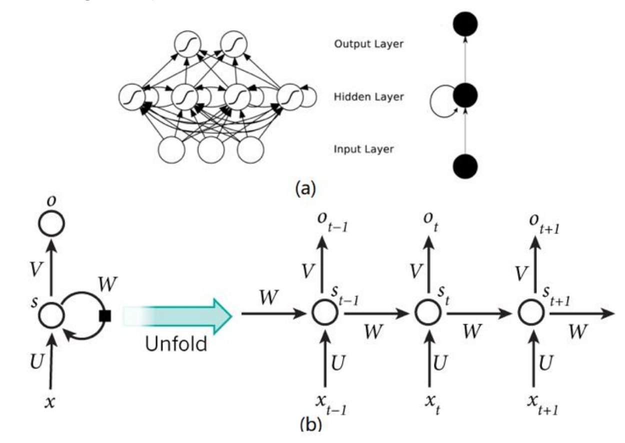


图 2　循环神经网络结构

循环神经网络最鲜明的特点是隐藏层之间存在权重连接，现有的输入与之前和后续存有关系。图 2(a)是一个典型的 RNNs 的连接情况示意图，体现了同一层之间的节点连接。RNNs 包含输入单元(Input units)，输入集标记为{x0,x1,...,xt,xt+1,...}，输出单元(Output units)的输出集则被标记为{y0,y1,...,yt,yt+1,..}。RNNs 还包含隐藏单元(Hidden units)，将其输出集标记为{s0,s1,...,st,st+1,...}，这些隐藏单元完成了最为主要的工作。图 2(b)所示是将循环神经网络进行展开成一个全神经网络，更便于人们理解。

模糊测试对象的输入数据是由各种的序列数据组成，数据之间存在上下文关系，如结构性的文档、各种格式文档数据排

列和解析，系统处理时按照预定的流程进行解析处理，这是选择运用循环神经网络来解决处理这类问题的出发点。

2.2　生成模型原理和描述

通过前文简述的测试用例生成方法对比，基于生成的方法效果较优，但时间和资源耗费代价高，基于变异的方法生成速度快，但效果欠佳。为此本文寻找折中的办法，提出了基于 RNNs 的生成模型。其基本方法原理是结合深度学习的优势构建特征提取算法和模型，让机器对有效的文本型数据进行学习训练，自动调整优化网络参数形成合适较优的处理模型参数，从而实现自动生成符合数据结构和规约的新数据，最后加入变异的规则和策略组合形成测试用例集合。通过该方式产生的数据和模糊测试的初衷比较契合，一方面较大限度地遵循被目标系统的基本规约，一方面要产生违反规约的数据触发异常。

本文选用字符文本型输入数据为研究对象，一是考虑到模型实现的复杂程度，目前计算机中常用的可见字符数为 98 个，较易于构建和训练网络；二是字符型数据做映射量少，相对需要的计算和存储资源较少，在常规运算能力下，相对计算资源和训练时间要求适中；三是目前网络应用发展较快 xml、json、字符文本型在实际应用也较广，在实际应用具备较高的性价比；四是虽然模型的输入是以字符、单词或易向量化的数据为对象，而随着深度学习的算法模型不断更新和降维方法的进步，分块处理或新的数据向量化方法，处理更多种类的数据扩展应用对象。

为此文本针对字符文本型数据构建基于循环神经网络的生成模型，实现学习、训练和预测生成新数据，最终结合变异策略组合产生新的测试用例。这里简要介绍生成模型的结构和功能模块。按图 3 所示分三层进行介绍。

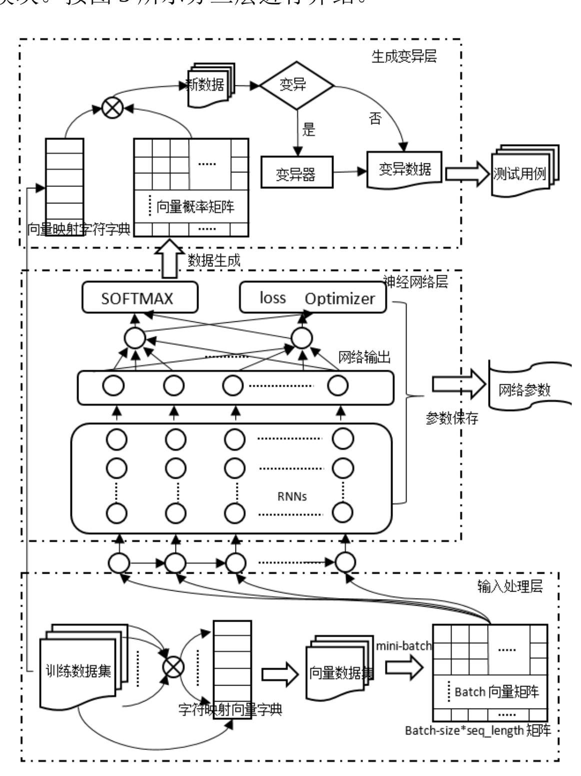


图 3　生成模型组成结构

2.2.1 输入处理层

该层将训练数据进行数据清理、转换等预处理后转换形成适合网络模型处理要求的数据类型和结构，创建 RNNs 输入空间，为训练提供数据输入源。其主要完成如下步骤：

a)生成映射向量表。为了把字符型的数据集合转码为适合于神经网络计算的向量数据，首先遍历训练数据集中的字符并生成唯一字符集，再选用合适的算法把字符集中的每个字符一对一的映射到数据向量，生成字符和数字向量相互转换映射表（即字符映射向量表和向量映射字符表），从而能够让字符和向量相互转换。常见的生成方法有 one-hot 编码、排序生成等。借鉴 Word2Vec 的原理，对复杂的数据集合需建模增加向量映射方法来进行关联计算，实现非字符型或字符组合型数据的映射处理，适应更多种类的训练数据。

b)训练数据集向量化。完成映射向量表产生后对输入的训练数据集进行向量化映射转换，把字符型集合编码转换为数字向量型集合，是对训练数据的向量化过程。

c)小批次(mini-batch)分割处理。因训练的数据集规模和计算处理能力差距，不能采用采用全数据集(full batch learning)的形式训练。考虑计算资源和神经网络构造参数选择合适的 batch-size 和 squence_length 对数据集合进行分批次训练。定义一个 batch 中的序列个数为 N(即 batch_size)，定义单个序列长度为 M(即为 squence_length)，每个 batch 就是 N×M 的数组。

d)创建数据输入空间。按 mini-batch 分割后每个 batch 的大小，建立 Shape=N×M 的 input 张量和 targets 张量，定义 keep_prob 空间用于控制 dropout 的保留节点数。

2.2.2 循环神经网络层

该层是生成模型的核心，对下接收输入层的数据，对上预测产生新数据。作为上下两层的衔接，数据运算处理分为网络训练和网络预测两个阶段。首先网络训练阶段，按照选取的网络和设定训练参数等构建循环神经网络，将输入层的数据输入网络迭代训练优化产生网络参数，并评估训练效果；然后网络预测阶段，加载初始数据和训练好的网络参数，通过循环神经网络预测产生新的概率矩阵输出至生成变异层。

影响该层的效果的最重要的部分是神经网络模型的选择和构建。自 RNNs 提出后，研究者们不断优化提出了多种复杂的 RNNs，这里着重介绍下使用最广泛、最成功的 LSTM 和 GRU 网络模型的基本情况和特点。

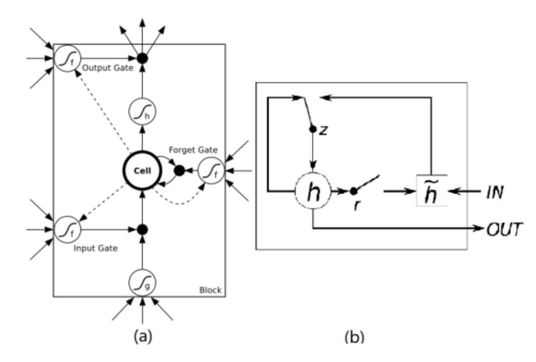


图 4　LSTM 和 GRU 模型结构

a)LSTM（long short-term memory）是一种特殊的 RNNs，1997 年 Hochreiter 和 Schmidhuber 最早提出，主要是为了解决长序列训练过程中的梯度消失和梯度爆炸问题。简单来说，就是相比普通的 RNN，LSTM 能够在更长的序列中有更好的表现。如图 4(a)所示，通过门控状态来控制传输状态，记住需要长时间记忆的，忘记不重要的信息，而不像普通的 RNN 那样只有一种记忆叠加方式，对很多需要"长期记忆"的任务来说尤其好用。但也因为引入了很多内容，导致参数变多，也使得训练难度加大了很多。

b)GRU（gate Recurrent Unit）与 LSTM 网络一样也是为了解决长期记忆和反向传播中的梯度等问题。GRU 输入输出的结构与普通的 RNN 相似，其中的内部思想与 LSTM 相似。与 LSTM 相比，GRU 内部少了一个"门控"，结构参见图 4(b)所示，参数比 LSTM 少，但是却也能够达到与 LSTM 相当的功能。

2.2.3 生成变异层

该层首先是由 RNNs 预测的新序列矩阵生成新数据，然后根据变异策略和算法对产生的数据进行必要的变异，最后按测试用例组装方法输出测试用例集。

a)数据生成的过程。在神经网络加载网络参数和初始数据后，网络能按学习好的参数、权重等不断循环预测新序列向量的概率矩阵，依据合适的策略，再把序列向量的概率矩阵映射为新的目标数据，通过这一过程实现了新数据的生成。这里调整 softmax 取样的温度(temperature)和引入随机数选择是影响生成新数据的常用办法。取样的温度变化会影响 RNNs 生成置信度，例如降低温度就会有更高的置信度，生成数据就更保守；相反，温度高，相关性越低数据就更多元化、更高的畸形度和更多可能触发异常。同理引入随机数选择预测概率最大的序列为新输出，也会增加生成数据的多样性。通过这些优化和调整方法，最终让网络生成的更多样和通过率较高的测试数据。

b)数据的变异过程。前面 RNNs 预测产生的数据除了具备相似的结构性也产生了一定的变异性，但受映射矩阵集合的限制，原则上难以产生超出映射集合的变异数据，因此引入如随机变异或其他针对性变异算法，可以增加变异度扩展测试用例的变异集合空间，从而让生成的测试用例触发更多的异常和覆盖更广的执行路径。

c)组装生成过程。考虑到可行性、计算资源和时间成本等因素，运用 RNNs 学习训练和产生的数据大多是不可以直接被加载执行的数据片段或未分割的数据块，需按测试对象对数据的处理规则和策略将新数据分割和组装产生可以执行的测试用例，最终输出形成测试用例集合。

3 实验和结果分析

为了验证循环神经网络生成模型在测试用例生成效果，本文选择了 CAJViewer7.2 阅读器和 PDF 文件为对象进行实验评估。让生成模型学习训练 PDF 文件中的字符文本数据，再由模型生成不同于已学习的数据，验证模型能否产生新数据，形成数据统计情况，评估 RNNs 训练差异和性能情况。选择构建对比测试用例集，评估生成的新用例在模糊测试过程中能否具备较高的覆盖率和通过率，以通过率和覆盖率来验证该方法模型的效果。

首先简述 CAJViewer7.2 和 PDF 文件格式的基本情况；其次结合 PDF 文件格式和模型特点提出实验方案；然后简述方案实施完成情况；最后以通过率和覆盖率这两项重要指标来分析了实验结果。

3.1 实验对象简述和设计分析

3.1.1 实验对象简述

a)CAJViewer 是中国期刊网的专用全文格式阅读器，支持中国期刊网的 TEB、CAJ、NH、KDH 和 PDF 格式文件阅读，有支持 MAC、iPad、iPhone、Android 平台的版本，使用较为广泛。这里选用 windows 版的 CAJViewer7.2 版完成以下实验。

b)PDF（portable document format，便携式文档格式）是由 Adobe Systems 用于与应用程序、操作系统、硬件无关的方式进行文件交换所发展出的文件格式。PDF 文件格式是实验的重要对象。这里简述 PDF 文件的基本结构。如图 5(a)所示的 PDF 文件大致可以分 4 个部分，即首部、文件体、交叉引用表和尾部。

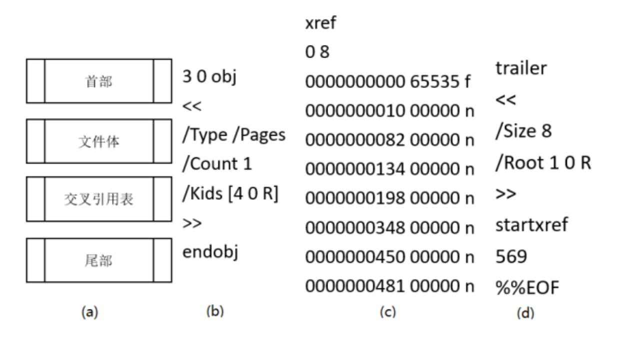


图 5 PDF 文件结构

PDF 文件结构的四个部分解释：

(a)首部，文件的开头，%PDF-1.7 的形式，其中最后一位就是 PDF 文件格式版本号，表示 PDF 文件遵循的版本规范，如"1.7"，目前最新的版本为 1.7。

(b)文件体，由若干个 obj 对象来组成，如图 5(b)所示，第一个数字是对象号，唯一标志一个对象；第二个是产生号，用来表明它在被创建后的第几次修改，新创建的产生号是 0。由关键字 obj 开始，对象的内容应该是包含在<<和>>之间的，最后以关键字 endobj 结束。包含二进制数据的 obj 内还会嵌套关键字 stream 和 endstream 来存放二进制数据。

(c)交叉引用表，如图 5(c)所示，用来索引各个 obj 对象在文档中的位置，以实现随机访问。Xref 标志一个交叉引用表的开始，表的第一行 0 8 说明下面各行所描述的对象号是从 0 开始，有 8 个对象。0000000000 65535 f，一般每个 PDF 文件都是以这一行开始交叉应用表的，对象 0 的起始地址为 0000000000，产生号（generation number）为 65535，也是最大产生号，不可以再进行更改，最后的 f 表明该对象为 free，这里

可以理解为是文件头。0000000009 00000 n 就是表示对象 1，0000000009 是其偏移地址，00000 为 5 位产生号，0 表明该对象未被修改过, n 表示该对象在使用，区别与自由对象(f)，可以更改。

(d)尾部，虽然放在文件的尾部，但这是文件内容逻辑的开头。如图 5(d)所示，/Size 8 说明该 PDF 文件的对象数目；/Root 1 0 R 说明根对象的对象号为 1；Startxref 553 说明交叉引用表的偏移地址，从而可以找到 PDF 文档中所有的对象的相对地址，进而访问对象。%%EOF 为文件结束标志。

3.1.2 分析和实现思路

a)理解文档的重点要素和关系。xref 是标定各个对象的地址，可以理解为对象的“目录”。trailer 是文档实际解析的文件头，通过重要的 Root 指向 Catalog 对象后，就相当于把接力棒交给了文档的 obj 对象。整个文档的内容结构是由以 ID 为指向标志的树状结构。以图 6 所示为例，首先 trailer 标定 root 指向了 Catalog 的 ID，再 Catalog 分支指向 Pages 和 Outline，再 Pages、Page、Resources 等，最终按 ID 指向序的链条延伸下去。可以理解 CAJViewer 阅读器处理的核心逻辑就是根据 obj 对象的指向不断发现和解析处理的循环。因此对 obj 对象分析理解是思考的重点。

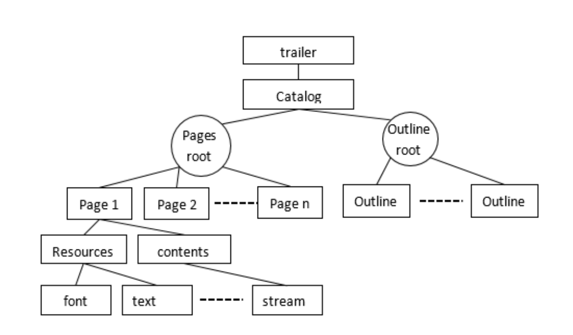


图 6　PDF 文件 obj 对象组织结构

b)分析 obj 对象情况。上述分析所知，PDF 文档结构的主体是由 obj 对象数据组成。Adobe 公司 2006 年 11 月发布的 pdf1.7 版格式文档中这部分的说明描述有大约 1 200 页，如果人工分析编写模糊测试模板变异生成文档存在以下难度，一是因 obj 对象种类和约规较多必定工作量巨大；二是对模版编写人员的计算机知识水平有较高的要求。通过对 obj 对象的分析可知除了 stream 部分是有二进制编码其余是可见字符编码，较适用于生成模型。

c)实践思考和设计。stream 部分的数据较为复杂，大多由外部扩展库实现处理，但受限于计算能力和时间成本等因素，因此把 PDF 文档分解为 obj 对象块来作为处理对象，设计了搜集 PDF 样本集、处理提取 obj 对象数据、RNNs 网络训练 obj 对象、RNNs 预测生成新 obj 对象、变异新 obj 对象、组装 obj 对象生成测试用例集和实验评估的实验步骤。

3.2　实验过程

依照上述思路，本文按 PDF 样本搜集和预处理、模型训练及分析、生成测试用例集和执行测试结果评估 4 个部分进行实验。

3.2.1 PDF 样本搜集和预处理

a)爬取 PDF 样本文件。互联网上发布有大量的各种类型的 PDF 文件，是一个较好的 PDF 样本获取源。为采集种类多样的 PDF 文件样本，以百度搜索引擎为入口，用“主题关键字+filetype:pdf”的组合的思路进行搜索 pdf 文件类型的链接。为绕过百度搜索引擎的反爬虫机制，这里使用 python+selenium2 技术，编写了基于浏览器的爬虫。综合互联网关键字排行和类别选取了 300 个主题关键字，运行 1 天时间爬取 PDF 文件 12 006 个。

b)PDF 样本集合整理。下载的 PDF 文件涉及的类型、种类较多并存在相似的文件类型，对获取的样本集合进行版本识别、分类和样本去重工作，获得有效的 PDF 文件 11 096 个，涵盖版本从 1.0～1.7 并按此进行分类，文件大小从 8 KB～43 MB。以 CAJViewer 为目标系统编写执行验证脚本和用 PeachMinset 进行最小化数据集精简筛选获得 1 149 个 PDF 文件的精简样本集。

c)训练数据集预处理。出于简化实验和适应模型数据集的目的，过滤了对包含“stream”二进制的数据，提取精简样本集中 PDF 文件的字符型的 obj 对象，如图 5(b)所示以“obj”开头到“endobj”结束。PDF 样本存在多种编码和字符集的问题，一一识别并处理花费时间较长，这里只提取了可显示的 ASCII 字符，然后计算其长度、md5 并去重后存入数据库，共计提取 316 991 条 obj 对象数据，长度从 11 字节～435 016 字节，长度区间分布如图 7(a)所示。经过分析数据后以长度分布为调准抽样选取 78 406 条数据，训练数据长度分布情况如图 7(b)所示，最终形成 14.6 MB 大小的训练数据集。

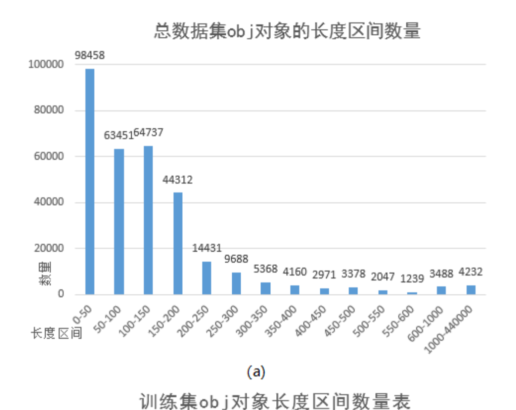


图 7　训练集 obj 对象长度分布情况

3.2.2 模型训练和分析

a)深度学习平台选择。TensorFlow 是谷歌 2015 年开源的基于 DistBelief 研发的第二代人工智能学习系统，将复杂的数据结构传输至人工智能神经网中进行分析和处理过程的系统。与 Caffe、Theano、Torch、MXNet 等框架相比，TensorFlow 在 Github 上 Fork 数和 Star 数都是最多的，而且在图形分类、音频处理、推荐系统和自然语言处理等场景下都有丰富的应用。综合以上原因，实验选用 Tensorflow 1.3 版构建神经网络层。

b)训练参数和硬件配置。为了对比训练效果选取了 LSTM 和 GRU 两种主要模型按 batch-size 为 20 和 50 组合形成 4 组参数，其他参数统一设置为序列长度 10、2 层、128 隐层节点、0.002 学习率和 100 轮次训练。硬件配置：Intel(R) Xeon(R) CPU E5-26xx v2，2 GB 内存的服务器 2 台。每一种模型 100 轮的训练时间大约为 30 h，训练的字符映射表字符数量为 98，生成 LSTM 模型参数大小为 3 125 436 字节，GRU 模型参数大小为 2 382 012 字节。训练参数和结果情况如表 1 所示。这里分别依照顺序记为 LSTM-20、LSTM-50、GRU-20 和 GRU-50。

表 1　模型训练对比

模型	分割	最佳	验证 ppl	用时/分/轮
LSTM	20	model-5114362	4.26923	18.8478
LSTM	50	model-2681565	3.791913	18.02646
GRU	20	model-6773074	5.322278	17.8175
GRU	50	model-884640	4.298033	16.26637

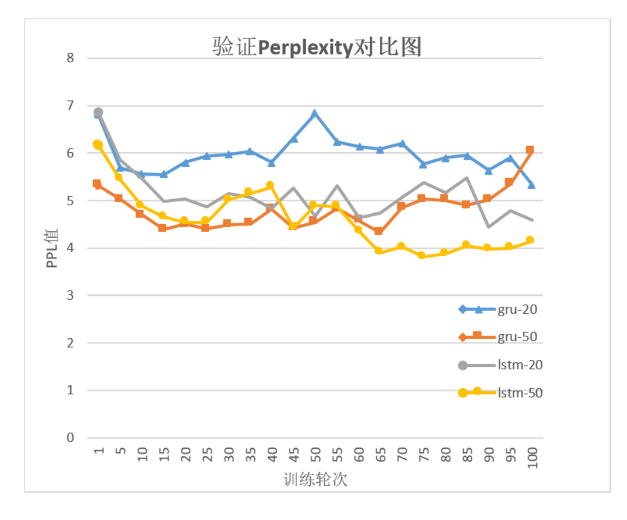


图 8　训练 Perplexity 对比

3.2.3 生成测试用例集

生成测试用例由 obj 对象生成、数据变异和数据组装这三个步骤。

a)obj 对象生成。为了对比分析分别将训练所得的最佳的 4 个模型参数加载到 RNNs 网络，让每个网络分别生成 10 000 个 obj 对象，共计 40 000 个，长度从 17～38 630 字节。如图 9 所示，可以看出网络生成的数据效果很好，已经能产生和训练集合中的单词和结构很相似的 obj 对象。通过把生成 obj 对象导入到数据库和总 obj 对象数据集做 md5 比对，未发现 md5 一致的两条数据，说明网络是能够自行生成新的不同的数据。

```
obj
<<
/Type /Font
/Subtype /Type1
/BaseFont /DY1255+ZIKB4C-1255
/FirstChar 33
/LastChar 56
/Widths [ 500 167 444 500 500 667 667 722 722 500 250 500 500 444 500 278 500
667 500 556 500 611 500 0 ]
/Encoding 6799 0 R
/FontDescriptor 9704 0 R
>>
endobj
```

图 9　循环神经网络生成的新 obj 对象

b)变异 obj 对象。

综合 obj 对象的结构特点，为了具备较好的通过率，这里只对“obj”与“endobj”之间数据进行变异，变异过程中用到了字符集变异和二进制变异。图 10 所示为用 python 编写的根据阀门值控制的随机变异算法。由 mut_frac 和 p 参数分别调整变异数量和两种变异方式的概率。图 11 所示为以 mut_frac=0.05 和 p=6 为参数进行变异的效果样例。

```python
def strRandomMutator(chars, str='Mutator', mut_frac=0.05, p=6):
    bytes_arr=bytearray(str, 'ascii')
    data_len = len(bytes_arr)
    # 按设定的变异率计算要变异的字符数量
    change_num = int(data_len*mut_frac)
    if data_len>change_num and data_len>18 and change_num>0:
        # 随机选择要变异的字符位置
        change_pos=random.sample(range(5,data_len-9),change_num)
        for pos in change_pos:
            # 加入随机因子来决定是字符集合或二进制变异
            if_rand = random.randint(0,10)
            if if_rand>p:# 二进制随机变异
                bytes_arr[pos] = random.randint(0,255)
            else:       # 字符集 变异
                bytes_arr[pos] = ord(random.sample(chars, 1)[0])
    ret_byte=bytes(bytes_arr)
    return ret_byte
```

图 10　随机变异算法

```
obj
<<
/Type 2Page
/Parent 4 0 9RS/MediaBnxq[0 0 6G2 79()
/Parent *3? 0 R
/ROsources <<
/Fo}t<<<
/F15 219 0 R
9>
:>
1<
/Parent 290 0 R
/Resources 98R0 Rl/Convents 328 0 R
>>
endobj
```

图 11　变异后的新 obj 对象

c)数据组装。由网络生成的数据是 obj 对象不能直接被 CAJViewer 打开执行，为了进行评估验证，就需要把 obj 对象组合成为 CAJViewer 可以解析的 PDF 文件。这里按照 PDF 文件基本结构分析创建了 7 个对象节点的 PDF 文件为用例生成模板，将变异后的 obj 对象替换更新 PDF 模板的 Catalog、Outlines、Pages、Page 和 Contents 部分，以用于测试评估 obj 对象在不同节点的效果。为了进行实验评估对比，创建了三种测试数据集：a)以用例生成模板 PDF 文件为样本用 Peach3 的随机变异方式产生 5 000 个测试用例，记为 PEACH 数据集；b)从之前提取的 obj 总数据集中随机抽取 1 000 个数据组装生成 5 000 个测试用例，记为 NORMAL 数据集；c)RNNs 生成的 obj 对象数据集中抽取 1 000 个 obj 对象按随机变异产生新的 1 000 个 obj 对象组装生成 5 000 个测试用例，记为 RNNS 数据集。

d)执行测试用例集。为评估测试用例集的效果和对比，在 Python 2.7 环境下编写了 CS 结构的测试用例执行工具，中心服务端存放测试用例集和负责分发测试用例，执行客户端安装 CAJViewer 执行环境，与服务端通信获取测试用例，分别执行用 Peachminset 产生 trace 文件和 pydbg 执行监控用于结果评估和分析。由 3 台服务器分别虚拟 1 台中心服务端和 10 台客户端完成执行测试。

3.2.4 结果分析

为验证和评估生成模型的可行性和生成的测试用例效果情况，本文针对选用的神经网络和测试用例来完成结果分析，进行了一是生成模型性能和可行性分析；二是对比评估分析测试用例效果。

1)生成模型性能和可行性分析

本次实验中选用了 LSTM 和 GRU 网络模型实验测试，选取其中最优的参数来生成的数据生成形式如图 9 所示的数据。从图中能直观地比较发现网络生成的数据和 PDF 样本文件中的数据结构基本一致，能正确地生成 obj 对象的开头、结尾、常用关键字和格式，只是在数据值上不同，通过 7 万条数据训练后，实际实验中用 4 个模型共测试共生成了 20 万条数据检验后未发现相同数据，数据长度也和训练数据集接近，说明了网络根据理解产生了相似的新数据，从触发异常概率角度来讲要优于随机变异的数据。性能情况，使用常规双核 26xx 系 CPU、2 GB 内存 Linux 服务器，训练 100 次 epoch 为例大约用时 30 个小时。可以从表 1 和图 8 所示验证 Perplexity (复杂度，是用来评价一个语言模型预测样本的标准，复杂度越低，代表模型的预测性能越好。)对比情况，LSTM-50 验证的 Perplexity 最佳，GRU-50 训练速度最快，不同的 batch-size 对训练效果有较大影响，与模型关于神经网络层的论述一致。

通过以上分析表明，除了以上两个模型仍有优化空间和更多模型选择可以提升效果，如注意力机制的 RNNs 模型等；该方法依赖于样本数据和向量化处理，综合实现代价和资源性能需求等问题，较适用于规约较为复杂的字符文本型数据；局限性学习训练效果依赖于数据样本，对于规约简单和缺乏可学习数据就不具太好优势和价值。

2)对比评估分析测试用例效果

为评估本实验中模型的效果，选取在模糊测试中最具代表性的代码覆盖率和通过率这两项指标来对比分析。

覆盖率方面，本文采用和覆盖率相同效果的动态链接库地址覆盖量来对比评估。由测试执行过程中 peachminset 执行 CAJViewer 生成的 trace 文件，编写统计程序，统计覆盖的动态链接库和地址情况。本文以统计 trace 文件的地址覆盖量为参照。覆盖量越多，表明程序覆盖的路径和代码块越多，则测试用例的效果就越好。

通过率方面，分析了 CAJViewer 阅读器对文件的处理过程，编写通过率验证程序，执行 CAJViewer 加载测试用例，监控执行过程中是否报错和弹出异常对话框，无任何异常为通过，反之为不通过。通过率和覆盖量对比如表 2 所示。

表 2　通过率和覆盖量对比

测试用例集	标记	覆盖量	总覆盖量	通过率
PEACH	peach-5000	151166	151166	52.80%
	Catalog-1000	192487		38.50%
	Outlines-1000	203151		56.50%
RNNS	Pages-1000	192673	236411	43.20%
	Page-1000	220387		65.10%
	Contents-1000	202106		68.50%
	Catalog-1000	181724		96.20%
	Outlines-1000	193530		97.40%
NORMAL	Pages-1000	182910	208621	97.60%
	Page-1000	207394		97.20%
	Contents-1000	193543		98.10%

由表 2 结果对比可知，PEACH 数据集是随机变异的产生，受初始样本和随机变异情况影响较大，变异速度较快但覆盖量较低，通过率较低，模糊测试发现漏洞的效率较低。RNNS 数据集的覆盖量最高，通过率较 NORMAL 数据集低，在通过率和覆盖率比较均衡，由循环神经网络产生的数据只有少量因结构规约被 CAJViewer 阅读器拒绝，鉴于模型自带的变异性和高覆盖量，因此具备触发程序异常的机会更多。NORMAL 数据集的通过率较高，覆盖量属中等但缺少数据变异，产生异常的可能性较低，相较于 RNNS 数据集存在覆盖量和变异度的较低。

以上分析表明生成模型能对字符文本型数据较为智能地形成用例生成基础规则，合理地生成能兼顾的通过率和覆盖率的测试用例，较适用于以字符文本型为表示基础的文件格式和通信协议的模糊测试的测试用例生成工作。

以上实验过程，除了验证了模型的可行性和优良效果，也通过实践过程启发了对于深度学习和模糊测试的进一步结合研究的方向，一是要扩大模型应用范围，需降低向量化的难度，可以考虑用卷积神经网络或有效的提取特征方法降维提高向量化能力；二是提升模型生成效果，可以借鉴 AFL 和 VUzzer 等有反馈能力指导等先进模糊测试思想，用神经网络对 trace 信息的学习和预测，由路径主导测试用例生成形成反馈闭环，提升路径探索深度和效率；三是研究合适的深度学习网络学习脆弱点特征和异常的导向分布等辅助权重知识，让测试用例生成更有针对性，更易触发异常。

4　结束语

本文简述了基于循环神经网络构造测试用例生成模型的基本原理和构架，以 CAJViewer 阅读器解析 PDF 文件为实验对象，进行了训练数据采集、数据处理、网络训练、数据生成组装和评估流程，结果分析验证了模型的可行性和实用性。经过由模糊测试、深度学习理论到模型设计和实践的过程，更能深刻体会深度学习网络解决复杂问题的优势，理解了依靠数据和网络模型为基础的特征提取分析处理方法，本质上是尽可能地代替

人工完成繁琐工作提升效率。同时，模糊测试技术中对特征的发掘和应用需求很多，深度学习与模糊测试之间存在较多的契合点，具备天然的结合优势。随着人工智能技术的发展和神经网络模型的层出不穷，可以预期必将会出现更多更优更好的算法和模型来提升模糊测试水平。

参考文献：

[1] 张雄，李舟军. 模糊测试技术研究综述 [J]. 计算机科学，2016, 43 (5): 1-8, 26. (Zhang Xiong, Li Zhoujun. Overview of fuzzy testing technology [J]. Computer Science, 2016, 43 (5): 1-8, 26.)

[2] 李彤，黄轩，黄睿. 模糊测试中测试用例生成方法 [J]. 计算机系统应用，2015, 24 (4): 139-143. (Li Tong, Huang Xuan, Huang Rui. Test case generation method in fuzzy testing [J]. Computer System Application, 2015, 24 (4): 139-143.)

[3] 李晓鹏. 文本表示算法的研究和应用 [D]. 北京：北京邮电大学，2016. (Li Xiaopeng. Research and application of text representation algorithm [D]. Beijing: Beijing University of Posts and Telecommunications, 2016.)

[4] Lipton Z C. A critical review of recurrent neural networks for sequence learning [C]// Proc of Computer Science. 2015: 1-35.

[5] Hochreiter S, Schmidhuber J. Long short-term memory [J]. Neural Computation, 1997, 9 (8): 1735-1780.

[6] Michael eddington, peach [EB/OL]. http://peachfuzzer. com.

[7] Godefroid P, Levin M Y, Molnar D. Sage: whitebox fuzzing for security testing [J]. Queue, 2012, 10 (1): 20.

[8] 黄磊，杜昌顺. 基于递归神经网络的文本分类研究 [J]. 北京化工大学学报：自然科学版，2017, 44 (1): 98-104. (Huang Lei, Du Changshun. Text classification based on recurrent neural network [J]. Journal of Beijing University of Chemical Technology: Natural Science Edition) , 2017, 44 (1): 98-104.)

[9] 李雪莲，段鸿，许牧. 基于 GRU 神经网络的中文分词法 [J//OL]. 厦门大学学报：自然科学版: 1-9. [2018-05-16]. (Li Xuelian, Duan Hong, Xu Mu. Chinese word segmentation method based on GRU neural network [J//OL]. Journal of Xiamen University: Natural Science Edition): 1-9. [2018-05-16].)

[10] Graves A, Schmidhuber J. Framewise phoneme classification with bidirectional LSTM and other neural network architectures [J]. Neural Networks, 2005, 18 (5): 602-610.

[11] Chung J, Gulcehre C, Cho K, *et al.* Gatedfeedback recurrent neural networks [C] //Proc of International Conference on Machine Learning. 2015.

[12] Pennington J, Socher R, Manning C D. GloVe: global vectors for word representation [C] //Proc of Conference on Empirical Methods in Natural Language Processing. 2014: 1532-1543.

[13] 张谦，高章敏，刘嘉勇. 基于 Word2vec 的微博短文本分类研究 [J]. 信息网络安全，2017 (1): 57-62. (Zhang Qian, Gao Zhangmin, Liu Jiayong. Research on micro-blog short text classification based on Word2vec [J]. Information Network Security, 2017 (1): 57-62.)

[14] 齐健，陈小明，游伟青. 基于 fuzzing 测试的网络协议安全评估方法研究 [J]. 信息网络安全，2017 (3): 59-65. (Qi Jian, Chen Xiaoming, You Weiqing. Research on network protocol security assessment method based on Fuzzing test [J]. Information Network Security, 2017 (3): 59-65.)

[15] 陈磊. 文本表示模型和特征选择算法研究 [D]. 合肥：中国科学技术大学，2017. (Chen Lei. Text representation model and feature selection algorithm [D]. Hefei: University of Science & Technology China, 2017.)

[16] 周练. Word2vec 的工作原理及应用探究 [J]. 科技情报开发与经济，2015, 25 (2): 145-148. (Zhou Lian. The working principle and application of Word2vec. [J]. Science and Technology Information Development and Economy, 2015, 25 (2): 145-148.)

[17] 郭丽丽，丁世飞. 深度学习研究进展 [J]. 计算机科学，2015, 42 (5): 28-33. (Guo Lili, Ding Shifei. Research progress in deep learning [J]. Computer Science, 2015, 42 (5): 28-33.)

[18] 欧阳永基，魏强，王嘉捷，等. 基于脆弱点特征导向的软件安全测试 [J]. 清华大学学报：自然科学版），2017, 57 (9): 903-908. (Ouyang Yongji, Wei Qiang, Wang Jiajie, *et al.* Software safety testing based on vulnerability feature oriented [J]. Journal of Tsinghua University: Natural Science Edition) , 2017, 57 (9): 903-908.)

[19] 欧阳永基，魏强，王清贤，等. 基于异常分布导向的智能 Fuzzing 方法 [J]. 电子与信息学报，2015, 37 (1): 143-149. (Ouyang Yongji, Wei Qiang, Wang Qingxian, *et al.* An intelligent Fuzzing method based on anomalous distribution oriented [J]. Journal of Electronics and Information, 2015, 37 (1): 143-149.)

[20] 张垚，张超容，林腾，等. 二进制代码测试覆盖率评估系统设计与实现 [J]. 指挥信息系统与技术，2015, 6 (6): 13-17. (Zhang Yao, Zhang Chaorong, Lin Teng, *et al.* Design and implementation of binary code test coverage evaluation system [J]. Command Information System and Technology, 2015, 6 (6): 13-17.)

[21] 伍海波. 基于神经网络的检测器生成算法研究与应用 [J]. 信息网络安全，2015 (9): 249-252. (Wu Haibo. Research and application of detector generation algorithm based on neural network [J]. Information Network Security, 2015 (9): 249-252.)

[22] 史记，曾昭龙，杨从保，等. Fuzzing 测试技术综述 [J]. 信息网络安全，2014 (3): 87-91. (Shi Ji, Zeng Zhaolong, Yang Congbao, *et al.* Test technology overview [J]. Information Network Security, 2014 (3): 87-91.)

[23] 章敏敏，徐和平，王晓洁，等. 谷歌 TensorFlow 机器学习框架及应用 [J]. 微型机与应用，2017, 36 (10): 58-60. (Zhang Minmin, Xu Heping, Wang Xiaojie, *et al.* Google tensor flow machine learning framework and application [J]. Microcomputers and Applications, 2017, 36 (10): 58-60.)

[24] Adobe Systems Incorporated. PDF reference, 6th edition, [EB/OL] (2006) . Available at http://www.adobe.com/content/dam/Adobe/en/devnet/acrobat/pdfs/pdf_reference_1-7. pdf.